Simplicial surfaces in GAP

Markus Baumeister
(j/w Alice Niemeyer)

Lehrstuhl B für Mathematik
RWTH Aachen University

30.08.2017

Basic data

- Package name: `SimplicialSurfaces`

Basic data

- Package name: `SimplicialSurfaces`
  - Not yet generally available

Basic data

- Package name: `SimplicialSurfaces`
  - Not yet generally available
- Authors: Alice Niemeyer, Markus Baumeister

Basic data

- Package name: `SimplicialSurfaces`
  - Not yet generally available
- Authors: Alice Niemeyer, Markus Baumeister
- based on current research at Lehrstuhl B including Plesken, Strzelczyk and others

Basic data

- Package name: `SimplicialSurfaces`
  - Not yet generally available
- Authors: Alice Niemeyer, Markus Baumeister
- based on current research at Lehrstuhl B including Plesken, Strzelczyk and others
- Internally used packages:

Basic data

- Package name: `SimplicialSurfaces`
  - Not yet generally available
- Authors: Alice Niemeyer, Markus Baumeister
- based on current research at Lehrstuhl B including Plesken, Strzelczyk and others
- Internally used packages:
  - `AttributeScheduler` by Gutsche

Basic data

- Package name: `SimplicialSurfaces`
  - Not yet generally available
- Authors: Alice Niemeyer, Markus Baumeister
- based on current research at Lehrstuhl B including Plesken, Strzelczyk and others
- Internally used packages:
  - `AttributeScheduler` by Gutsche
  - `Digraphs` by De Beule, Mitchell, Pfeiffer, Wilson et al.

Basic data

- Package name: `SimplicialSurfaces`
  - Not yet generally available
- Authors: Alice Niemeyer, Markus Baumeister
- based on current research at Lehrstuhl B including Plesken, Strzelczyk and others
- Internally used packages:
  - `AttributeScheduler` by Gutsche
  - `Digraphs` by De Beule, Mitchell, Pfeiffer, Wilson et al.
  - `GAPDoc` by Lübeck

Basic data

- Package name: `SimplicialSurfaces`
  - Not yet generally available
- Authors: Alice Niemeyer, Markus Baumeister
- based on current research at Lehrstuhl B including Plesken, Strzelczyk and others
- Internally used packages:
  - `AttributeScheduler` by Gutsche
  - `Digraphs` by De Beule, Mitchell, Pfeiffer, Wilson et al.
  - `GAPDoc` by Lübeck
  - `AutoDoc` by Gutsche

Motivation

Motivation

Goal: Investigate paper folding

Motivation

Goal: Investigate paper folding

- rigid folding in $\mathbb{R}^3$

Motivation

Goal: Investigate paper folding

- rigid folding in $\mathbb{R}^3$
- consider surfaces built from triangles (simplicial surfaces)

Motivation

Goal: Investigate paper folding

- rigid folding in $\mathbb{R}^3$
- consider surfaces built from triangles (simplicial surfaces)
    - not closed under folding

Motivation

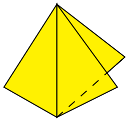
Goal: Investigate paper folding

- rigid folding in $\mathbb{R}^3$
- consider surfaces built from triangles (simplicial surfaces)
    - not closed under folding
    - allow more general structures:

Motivation

Goal: Investigate paper folding

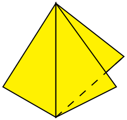
- rigid folding in $\mathbb{R}^3$
- consider surfaces built from triangles (simplicial surfaces)
  - not closed under folding
  - allow more general structures:

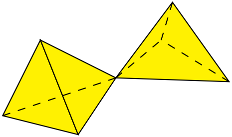


Motivation

Goal: Investigate paper folding

- rigid folding in $\mathbb{R}^3$
- consider surfaces built from triangles (simplicial surfaces)
  - not closed under folding
  - allow more general structures:

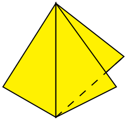


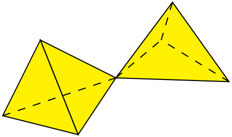


Motivation

Goal: Investigate paper folding

- rigid folding in $\mathbb{R}^3$
- consider surfaces built from triangles (simplicial surfaces)
  - not closed under folding
  - allow more general structures:





- embeddings are difficult to compute

Motivation

Goal: Investigate paper folding

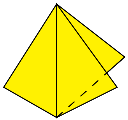
- rigid folding in $\mathbb{R}^3$
- consider surfaces built from triangles (simplicial surfaces)
    - not closed under folding
    - allow more general structures:

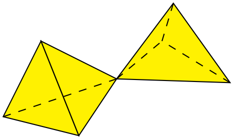
- embeddings are difficult to compute
    - some embeddings of asymmetric icosahedrons are infeasible to compute

Motivation

Goal: Investigate paper folding

- rigid folding in $\mathbb{R}^3$
- consider surfaces built from triangles (simplicial surfaces)
  - not closed under folding
  - allow more general structures:





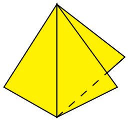
- embeddings are difficult to compute
  - some embeddings of asymmetric icosahedrons are infeasible to compute

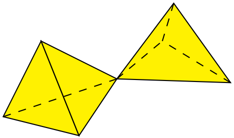
$\rightsquigarrow$ focus on intrinsic properties

Motivation

Goal: Investigate paper folding

- rigid folding in $\mathbb{R}^3$
- consider surfaces built from triangles (simplicial surfaces)
  - not closed under folding
  - allow more general structures:





- embeddings are difficult to compute
  - some embeddings of asymmetric icosahedrons are infeasible to compute

$\rightsquigarrow$ focus on intrinsic properties

$\rightsquigarrow$ incidence geometry

Implementation in GAP

Implementation in GAP

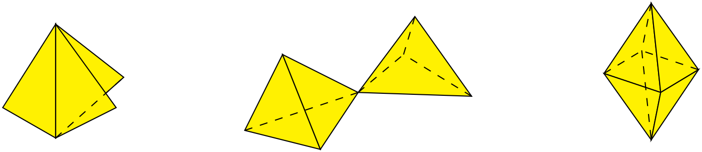
- can describe incidence geometry

Implementation in GAP

- can describe incidence geometry
  - allows flexible access to the incidence geometry (`AttributeScheduler`)

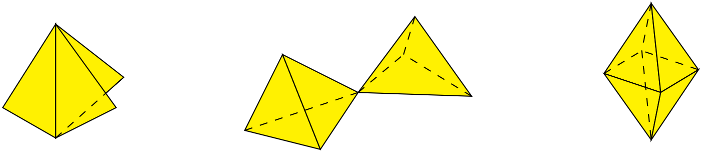
Implementation in GAP

- can describe incidence geometry
  - allows flexible access to the incidence geometry (`AttributeScheduler`)
- can manage hierarchy of structures



Implementation in GAP

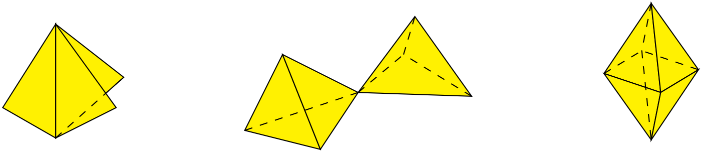
- can describe incidence geometry
  - allows flexible access to the incidence geometry (`AttributeScheduler`)
- can manage hierarchy of structures



- works well with group–theoretic descriptions

Implementation in GAP

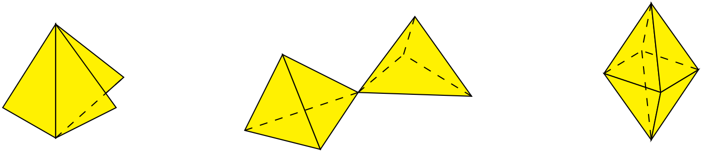
- can describe incidence geometry
  - allows flexible access to the incidence geometry (`AttributeScheduler`)
- can manage hierarchy of structures



- works well with group–theoretic descriptions
- difference to `FinInG`–package by De Beule, Neunhöffer et al.

Implementation in GAP

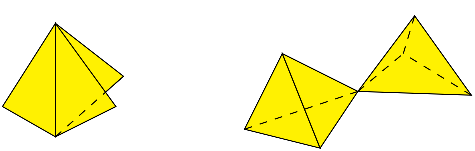
- can describe incidence geometry
  - allows flexible access to the incidence geometry (`AttributeScheduler`)
- can manage hierarchy of structures



- works well with group–theoretic descriptions
- difference to `FinInG`–package by De Beule, Neunhöffer et al.
  - only two dimensions

Implementation in GAP

- can describe incidence geometry
  - allows flexible access to the incidence geometry (`AttributeScheduler`)
- can manage hierarchy of structures



- works well with group–theoretic descriptions
- difference to `FinInG`–package by De Beule, Neunhöffer et al.
  - only two dimensions but it can work with colourings and foldings

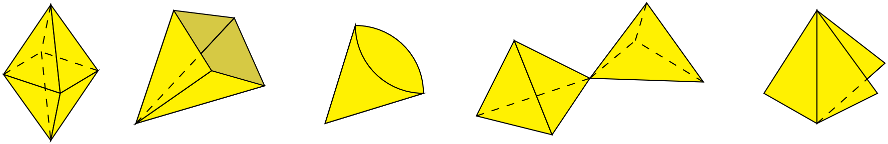
1. General simplicial surfaces

1. General simplicial surfaces

2. Edge colouring and group properties

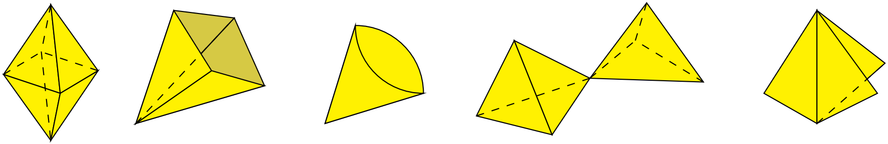
We want to describe different structures:

We want to describe different structures:



Triangular complexes

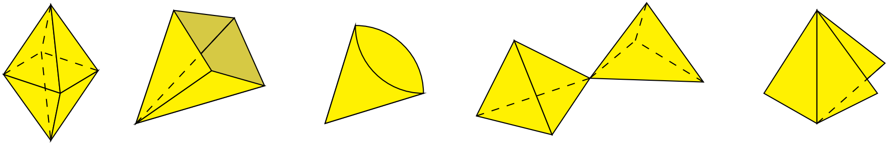
We want to describe different structures:



$\rightsquigarrow$ **triangular complexes**

Triangular complexes

We want to describe different structures:

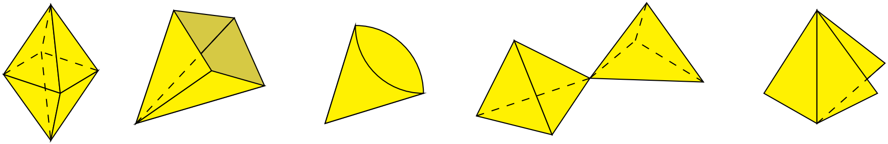


$\rightsquigarrow$ **triangular complexes**

- sets of vertices, edges and faces

Triangular complexes

We want to describe different structures:

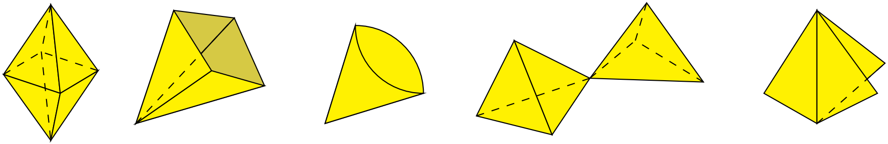


$\rightsquigarrow$ **triangular complexes**

- sets of vertices, edges and faces
- incidence relation between them

Triangular complexes

We want to describe different structures:

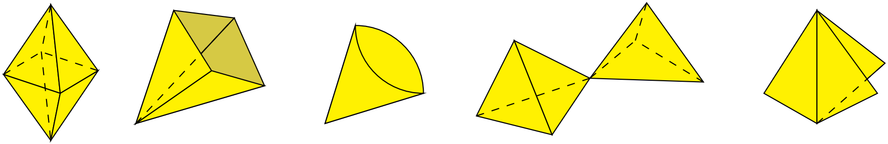


$\rightsquigarrow$ **triangular complexes**

- sets of vertices, edges and faces
- incidence relation between them
- every face is a triangle

Triangular complexes

We want to describe different structures:

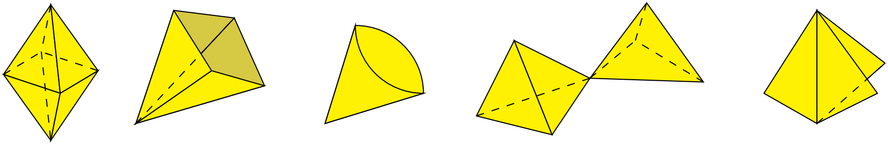


$\rightsquigarrow$ **triangular complexes**

- sets of vertices, edges and faces
- incidence relation between them
- every face is a triangle
- every vertex lies in an edge

Triangular complexes

We want to describe different structures:



$\rightsquigarrow$ **triangular complexes**

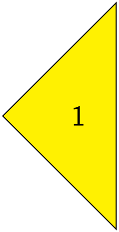
- sets of vertices, edges and faces
- incidence relation between them
- every face is a triangle
- every vertex lies in an edge and every edge lies in a face

Isomorphism testing

Incidence structures can be interpreted as coloured graphs:

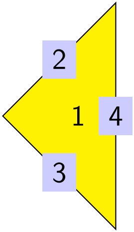
Isomorphism testing

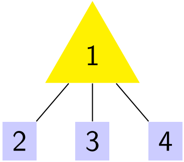
Incidence structures can be interpreted as coloured graphs:



Isomorphism testing

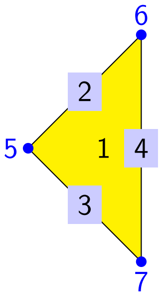
Incidence structures can be interpreted as coloured graphs:

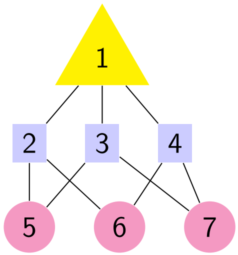




Isomorphism testing

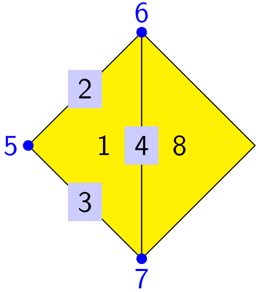
Incidence structures can be interpreted as coloured graphs:

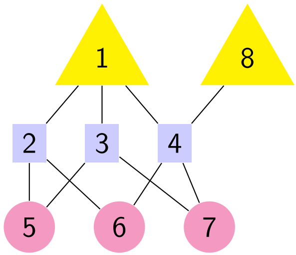




Isomorphism testing

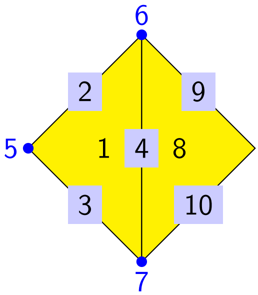
Incidence structures can be interpreted as coloured graphs:

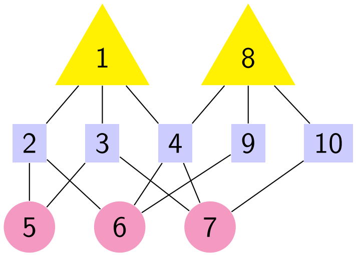




Isomorphism testing

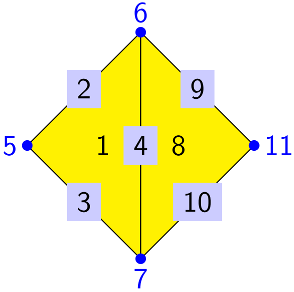
Incidence structures can be interpreted as coloured graphs:

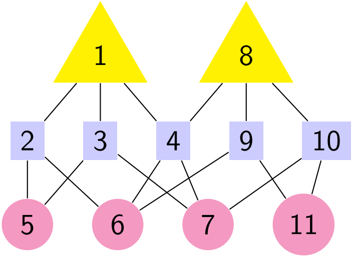




Isomorphism testing

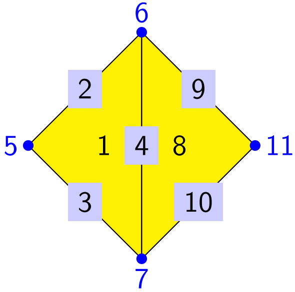
Incidence structures can be interpreted as coloured graphs:

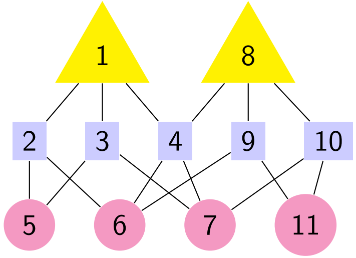




Isomorphism testing

Incidence structures can be interpreted as coloured graphs:

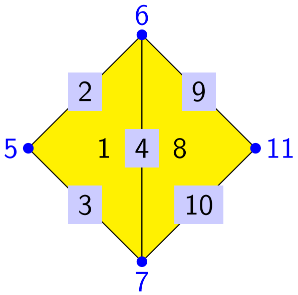


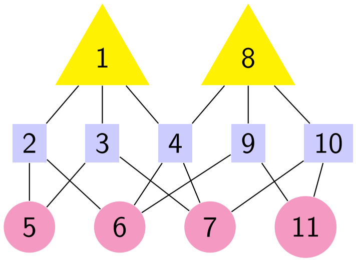


$\rightsquigarrow$ reduce to graph isomorphism problem

Isomorphism testing

Incidence structures can be interpreted as coloured graphs:



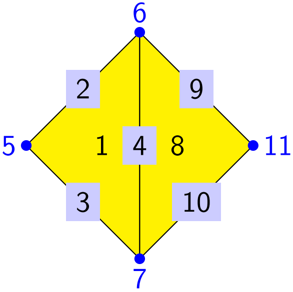


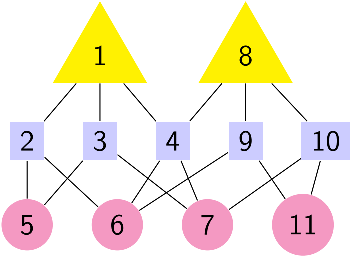
$\rightsquigarrow$ reduce to graph isomorphism problem

$\rightsquigarrow$ solved efficiently in practice by `nauty/Traces` (McKay, Piperno)

Isomorphism testing

Incidence structures can be interpreted as coloured graphs:

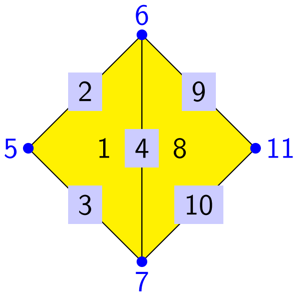


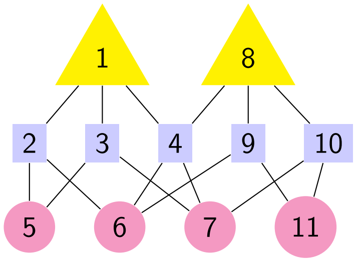


$\rightsquigarrow$ reduce to graph isomorphism problem
$\rightsquigarrow$ solved efficiently in practice by `nauty/Traces` (McKay, Piperno)
In GAP: `NautyTracesInterface` (Gutsche, Niemeyer, Schweitzer)

Isomorphism testing

Incidence structures can be interpreted as coloured graphs:





$\rightsquigarrow$ reduce to graph isomorphism problem

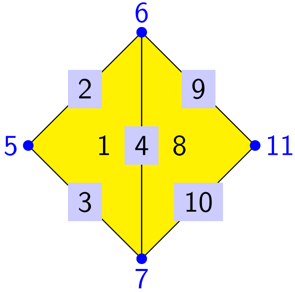
$\rightsquigarrow$ solved efficiently in practice by `nauty/Traces` (McKay, Piperno)

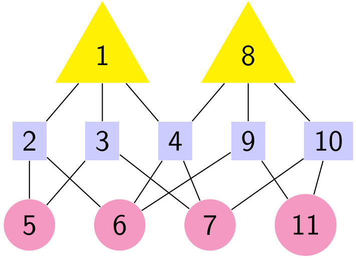
In GAP: `NautyTracesInterface` (Gutsche, Niemeyer, Schweitzer)

- calls C–functions directly without writing files

Isomorphism testing

Incidence structures can be interpreted as coloured graphs:





$\rightsquigarrow$ reduce to graph isomorphism problem

$\rightsquigarrow$ solved efficiently in practice by `nauty/Traces` (McKay, Piperno)

In GAP: `NautyTracesInterface` (Gutsche, Niemeyer, Schweitzer)

- calls C–functions directly without writing files
- also returns automorphism group

General properties

Some properties can be computed for all triangular complexes:

General properties

Some properties can be computed for all triangular complexes:

- Connectivity

General properties

Some properties can be computed for all triangular complexes:

- Connectivity
- Euler–Characteristic

General properties

Some properties can be computed for all triangular complexes:

- Connectivity
- Euler–Characteristic

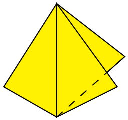
Orientability is **not** one of them.

General properties

Some properties can be computed for all triangular complexes:

- Connectivity
- Euler–Characteristic

Orientability is **not** one of them. Counterexample:

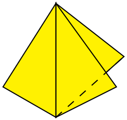


General properties

Some properties can be computed for all triangular complexes:

- Connectivity
- Euler–Characteristic

Orientability is **not** one of them. Counterexample:



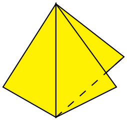
$\Rightarrow$ every edge lies in at most two faces (for well–definedness)

General properties

Some properties can be computed for all triangular complexes:

- Connectivity
- Euler–Characteristic

Orientability is **not** one of them. Counterexample:

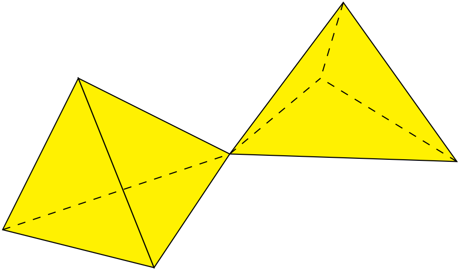


$\Rightarrow$ every edge lies in at most two faces (for well–definedness)
$\rightsquigarrow$ **ramified simplicial surfaces**

Why ramified?

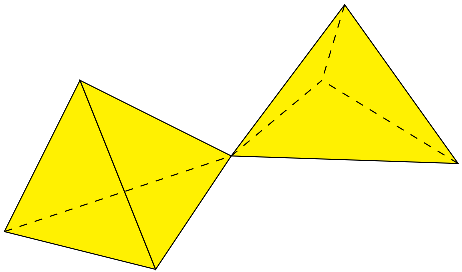
Why ramified?

Typical example of a ramified simplicial surface:



Why ramified?

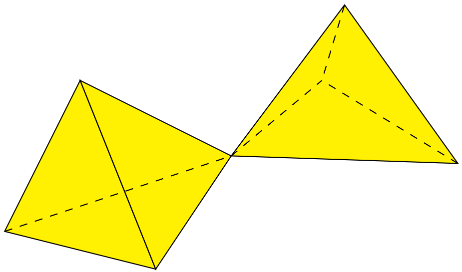
Typical example of a ramified simplicial surface:



$\Rightarrow$ It is not a surface – there is a *ramification* at the central vertex

Why ramified?

Typical example of a ramified simplicial surface:



$\Rightarrow$ It is not a surface – there is a *ramification* at the central vertex
A **simplicial surface** does not have these ramifications.

Classification

Classification

Plesken/Strzelczyk classified all closed simplicial surfaces up to 20 triangles.

Classification

Plesken/Strzelczyk classified all closed simplicial surfaces up to 20 triangles.

- building blocks are those without a 3–cycle of edges

Classification

Plesken/Strzelczyk classified all closed simplicial surfaces up to 20 triangles.

- building blocks are those without a 3–cycle of edges
- e. g. there are 87 non–isomorphic surfaces with 20 triangles

Classification

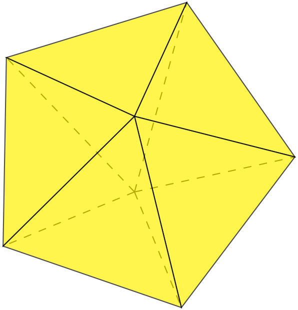
Plesken/Strzelczyk classified all closed simplicial surfaces up to 20 triangles.

- building blocks are those without a 3–cycle of edges
- e. g. there are 87 non–isomorphic surfaces with 20 triangles
- e. g. there is only one surface with 10 triangles:

Classification

Plesken/Strzelczyk classified all closed simplicial surfaces up to 20 triangles.

- building blocks are those without a 3–cycle of edges
- e. g. there are 87 non–isomorphic surfaces with 20 triangles
- e. g. there is only one surface with 10 triangles:



Progress report of triangulated complexes

Progress report of triangulated complexes

Already implemented:

Progress report of triangulated complexes

Already implemented:

- surface hierarchy

Progress report of triangulated complexes

Already implemented:

- surface hierarchy
- elementary properties (e. g. connectivity, orientability)

Progress report of triangulated complexes

Already implemented:

- surface hierarchy
- elementary properties (e. g. connectivity, orientability)
- isomorphism testing

Progress report of triangulated complexes

Already implemented:

- surface hierarchy
- elementary properties (e. g. connectivity, orientability)
- isomorphism testing
- classification of small surfaces (as data base)

Progress report of triangulated complexes

Already implemented:

- surface hierarchy
- elementary properties (e. g. connectivity, orientability)
- isomorphism testing
- classification of small surfaces (as data base)

Not yet implemented:

Progress report of triangulated complexes

Already implemented:

- surface hierarchy
- elementary properties (e. g. connectivity, orientability)
- isomorphism testing
- classification of small surfaces (as data base)

Not yet implemented:

- automorphism group

Progress report of triangulated complexes

Already implemented:

- surface hierarchy
- elementary properties (e. g. connectivity, orientability)
- isomorphism testing
- classification of small surfaces (as data base)

Not yet implemented:

- automorphism group
- advanced properties (any wishes?)

Embedding questions

Embedding questions

Given: A triangular complex

Embedding questions

Given: A triangular complex

- Can it be embedded into $\mathbb{R}^3$?

Embedding questions

Given: A triangular complex

- Can it be embedded into $\mathbb{R}^3$?
- In how many ways?

Embedding questions

Given: A triangular complex

- Can it be embedded into $\mathbb{R}^3$?
- In how many ways?

Simplifications:

Embedding questions

Given: A triangular complex

- Can it be embedded into $\mathbb{R}^3$?
- In how many ways?

Simplifications:

1. Only simplicial surfaces (that are built from triangles)

Embedding questions

Given: A triangular complex

- Can it be embedded into $\mathbb{R}^3$?
- In how many ways?

Simplifications:

1. Only simplicial surfaces (that are built from triangles)
2. All triangles are isometric

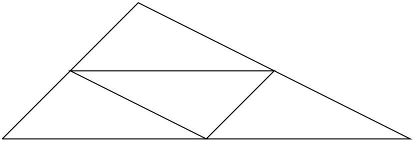
Embedding questions

Given: A triangular complex

- Can it be embedded into $\mathbb{R}^3$?
- In how many ways?

Simplifications:

1. Only simplicial surfaces (that are built from triangles)
2. All triangles are isometric



Embedding questions

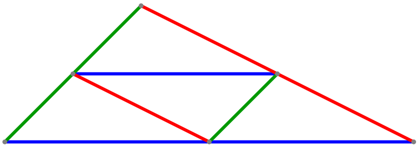
Given: A triangular complex

- Can it be embedded into $\mathbb{R}^3$?
- In how many ways?

Simplifications:

1. Only simplicial surfaces (that are built from triangles)
2. All triangles are isometric

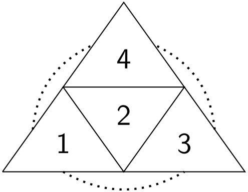
$\rightsquigarrow$ Edge–colouring encodes different lengths



Colouring as permutation

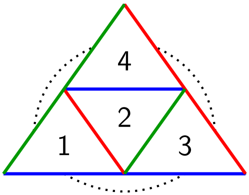
Colouring as permutation

Consider tetrahedron



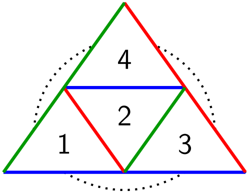
Colouring as permutation

Consider tetrahedron with edge colouring



Colouring as permutation

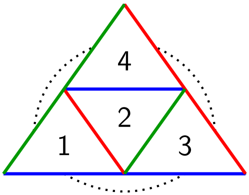
Consider tetrahedron with edge colouring



simplicial surface $\Rightarrow$

Colouring as permutation

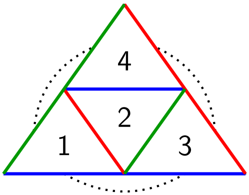
Consider tetrahedron with edge colouring



simplicial surface $\Rightarrow$ at most two faces at each edge

Colouring as permutation

Consider tetrahedron with edge colouring

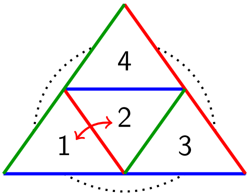


simplicial surface $\Rightarrow$ at most two faces at each edge

$\rightsquigarrow$ every edge defines a transposition of incident faces

Colouring as permutation

Consider tetrahedron with edge colouring



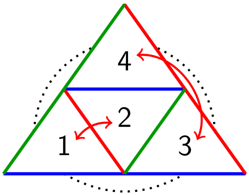
simplicial surface $\Rightarrow$ at most two faces at each edge

$\rightsquigarrow$ every edge defines a transposition of incident faces

- (1,2)

Colouring as permutation

Consider tetrahedron with edge colouring



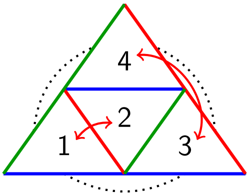
simplicial surface $\Rightarrow$ at most two faces at each edge

$\rightsquigarrow$ every edge defines a transposition of incident faces

- (1,2)(3,4)

Colouring as permutation

Consider tetrahedron with edge colouring



simplicial surface $\Rightarrow$ at most two faces at each edge

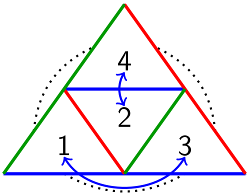
$\rightsquigarrow$ every edge defines a transposition of incident faces

$\rightsquigarrow$ every colour class defines a permutation of the faces

- (1,2)(3,4)

Colouring as permutation

Consider tetrahedron with edge colouring

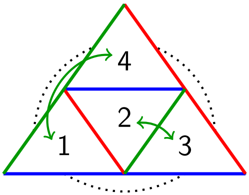


simplicial surface $\Rightarrow$ at most two faces at each edge

- $\rightsquigarrow$ every edge defines a transposition of incident faces
- $\rightsquigarrow$ every colour class defines a permutation of the faces
  - (1,2)(3,4) , (1,3)(2,4)

Colouring as permutation

Consider tetrahedron with edge colouring



simplicial surface $\Rightarrow$ at most two faces at each edge

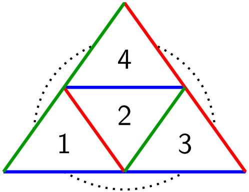
$\rightsquigarrow$ every edge defines a transposition of incident faces

$\rightsquigarrow$ every colour class defines a permutation of the faces

- (1,2)(3,4) , (1,3)(2,4) , (1,4)(2,3)

Colouring as permutation

Consider tetrahedron with edge colouring

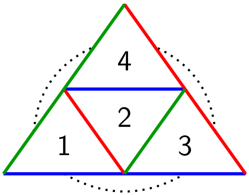


simplicial surface $\Rightarrow$ at most two faces at each edge

- $\rightsquigarrow$ every edge defines a transposition of incident faces
- $\rightsquigarrow$ every colour class defines a permutation of the faces
  - (1,2)(3,4) , (1,3)(2,4) , (1,4)(2,3)
- $\rightsquigarrow$ group theoretic considerations

Colouring as permutation

Consider tetrahedron with edge colouring

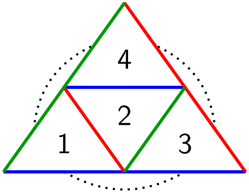


simplicial surface $\Rightarrow$ at most two faces at each edge

- $\rightsquigarrow$ every edge defines a transposition of incident faces
- $\rightsquigarrow$ every colour class defines a permutation of the faces
  - (1,2)(3,4) , (1,3)(2,4) , (1,4)(2,3)
- $\rightsquigarrow$ group theoretic considerations
  - The connected components of the surface correspond to

Colouring as permutation

Consider tetrahedron with edge colouring



simplicial surface $\Rightarrow$ at most two faces at each edge

$\rightsquigarrow$ every edge defines a transposition of incident faces

$\rightsquigarrow$ every colour class defines a permutation of the faces

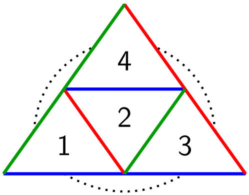
- (1,2)(3,4) , (1,3)(2,4) , (1,4)(2,3)

$\rightsquigarrow$ group theoretic considerations

- The connected components of the surface correspond to the orbits of $\langle \sigma_a, \sigma_b, \sigma_c \rangle$ on the faces

Colouring as permutation

Consider tetrahedron with edge colouring



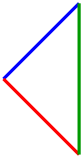
simplicial surface $\Rightarrow$ at most two faces at each edge

- $\rightsquigarrow$ every edge defines a transposition of incident faces
- $\rightsquigarrow$ every colour class defines a permutation of the faces
  - (1,2)(3,4) , (1,3)(2,4) , (1,4)(2,3)
- $\rightsquigarrow$ group theoretic considerations
  - The connected components of the surface correspond to the orbits of $\langle \sigma_a, \sigma_b, \sigma_c \rangle$ on the faces (fast computation for permutation groups)

How do faces fit together?

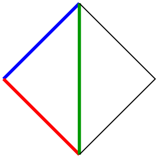
How do faces fit together?

Consider a face of the surface



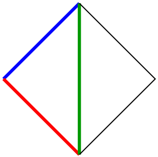
How do faces fit together?

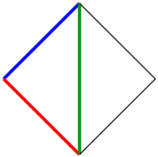
Consider a face of the surface and a neighbouring face.



How do faces fit together?

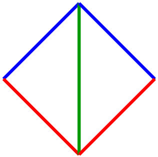
Consider a face of the surface and a neighbouring face.
The neighbour can be coloured in two ways:

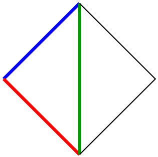




How do faces fit together?

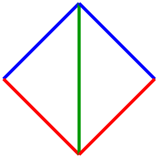
Consider a face of the surface and a neighbouring face.
The neighbour can be coloured in two ways:

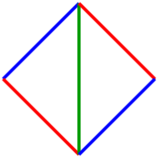




How do faces fit together?

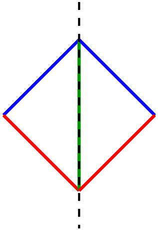
Consider a face of the surface and a neighbouring face.
The neighbour can be coloured in two ways:



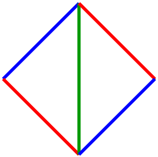


How do faces fit together?

Consider a face of the surface and a neighbouring face.
The neighbour can be coloured in two ways:

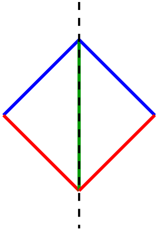


mirror (m)

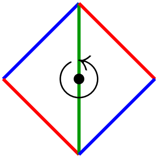


How do faces fit together?

Consider a face of the surface and a neighbouring face.
The neighbour can be coloured in two ways:



mirror (m)



rotation (r)

How do faces fit together?

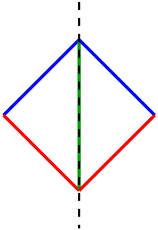
Consider a face of the surface and a neighbouring face.
The neighbour can be coloured in two ways:

mirror (m) rotation (r)

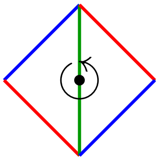
This gives an **mr–assignment** for the edges of the surface.

How do faces fit together?

Consider a face of the surface and a neighbouring face.
The neighbour can be coloured in two ways:



mirror (m)



rotation (r)

This gives an **mr–assignment** for the edges of the surface.

Lemma

Permutations and mr–assignment uniquely determine the surface.

Constructing surfaces from groups

Constructing surfaces from groups

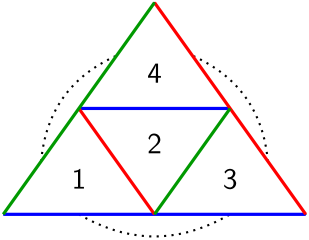
A general mr–assignment leads to complicated surfaces.

Constructing surfaces from groups

A general mr–assignment leads to complicated surfaces.
Simplification: edges of same colour have the same mr–type

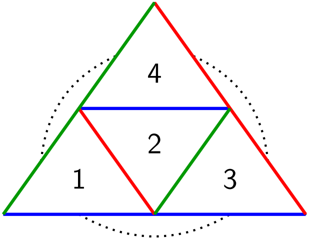
Constructing surfaces from groups

A general mr–assignment leads to complicated surfaces.
Simplification: edges of same colour have the same mr–type
Example



Constructing surfaces from groups

A general mr–assignment leads to complicated surfaces.
Simplification: edges of same colour have the same mr–type
Example



has only r–edges.

The mirror–case

The mirror–case

If all edges are mirrors, the situation is simple.

The mirror–case

If all edges are mirrors, the situation is simple.

Lemma

A simplicial surface has only mirror–edges iff it covers a single triangle

The mirror–case

If all edges are mirrors, the situation is simple.

Lemma

A simplicial surface has only mirror–edges iff it covers a single triangle, i. e. there is a surjective incidence–preserving map

The mirror–case

If all edges are mirrors, the situation is simple.

Lemma

A simplicial surface has only mirror–edges iff it covers a single triangle, i. e. there is a surjective incidence–preserving map to the simplicial surface consisting of exactly one face.

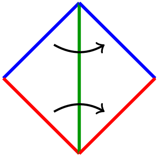
The mirror–case

If all edges are mirrors, the situation is simple.

Lemma

A simplicial surface has only mirror–edges iff it covers a single triangle, i. e. there is a surjective incidence–preserving map to the simplicial surface consisting of exactly one face.

Consider



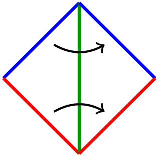
The mirror–case

If all edges are mirrors, the situation is simple.

Lemma

A simplicial surface has only mirror–edges iff it covers a single triangle, i. e. there is a surjective incidence–preserving map to the simplicial surface consisting of exactly one face.

Consider



$\Rightarrow$ Unique map that preserves incidence

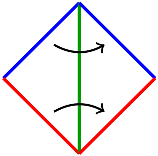
The mirror–case

If all edges are mirrors, the situation is simple.

Lemma

A simplicial surface has only mirror–edges iff it covers a single triangle, i. e. there is a surjective incidence–preserving map to the simplicial surface consisting of exactly one face.

Consider



$\Rightarrow$ Unique map that preserves incidence

- Covering pulls back a mirror–colouring of the triangle.

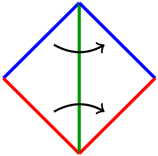
The mirror–case

If all edges are mirrors, the situation is simple.

Lemma

A simplicial surface has only mirror–edges iff it covers a single triangle, i. e. there is a surjective incidence–preserving map to the simplicial surface consisting of exactly one face.

Consider



$\Rightarrow$ Unique map that preserves incidence

- Covering pulls back a mirror–colouring of the triangle.
- Mirror–colouring defines a map to the triangle.

Construction from permutations

Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation

Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation (like generators of a finite group)

Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation (like generators of a finite group)

Lemma

There exists a coloured surface with the given involutions

Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation (like generators of a finite group)

Lemma

There exists a coloured surface with the given involutions where all edges are mirror edges.

Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation (like generators of a finite group)

Lemma

There exists a coloured surface with the given involutions where all edges are mirror edges.

- The faces are the points moved by the involution triple

Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation (like generators of a finite group)

Lemma

There exists a coloured surface with the given involutions where all edges are mirror edges.

- The faces are the points moved by the involution triple
- The edges are the 1- and 2–cycles of the involutions

Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation (like generators of a finite group)

Lemma

There exists a coloured surface with the given involutions where all edges are mirror edges.

- The faces are the points moved by the involution triple
- The edges are the 1- and 2–cycles of the involutions
- The vertices are

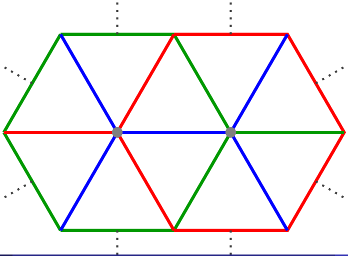
Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation (like generators of a finite group)

Lemma

There exists a coloured surface with the given involutions where all edges are mirror edges.

- The faces are the points moved by the involution triple
- The edges are the 1- and 2–cycles of the involutions
- The vertices are



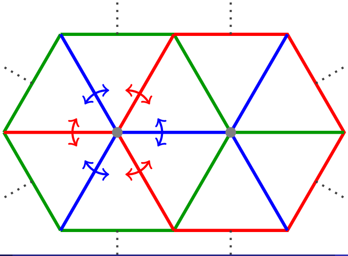
Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation (like generators of a finite group)

Lemma

There exists a coloured surface with the given involutions where all edges are mirror edges.

- The faces are the points moved by the involution triple
- The edges are the 1- and 2–cycles of the involutions
- The vertices are



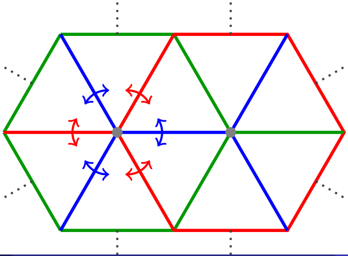
Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation (like generators of a finite group)

Lemma

There exists a coloured surface with the given involutions where all edges are mirror edges.

- The faces are the points moved by the involution triple
- The edges are the 1- and 2–cycles of the involutions
- The vertices are the orbits of $\langle \sigma_a, \sigma_b \rangle$ on the faces



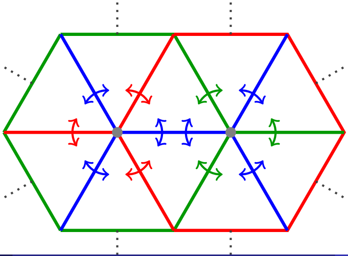
Construction from permutations

Start with three involutions σ_a, σ_b, σ_c in permutation representation (like generators of a finite group)

Lemma

There exists a coloured surface with the given involutions where all edges are mirror edges.

- The faces are the points moved by the involution triple
- The edges are the 1- and 2–cycles of the involutions
- The vertices are the orbits of $\langle \sigma_a, \sigma_b \rangle$ on the faces (for all pairs)



Construction example

Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$

Construction example

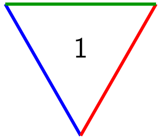
$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$

Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$

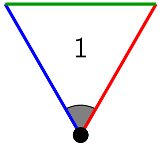
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



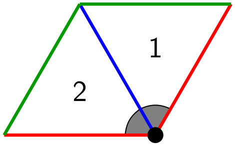
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



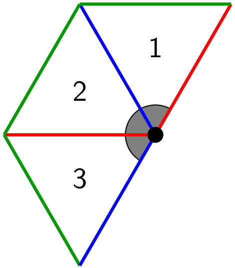
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



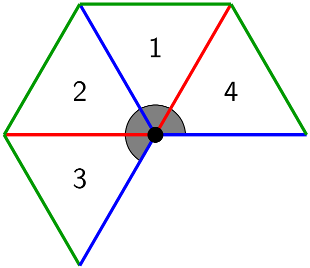
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



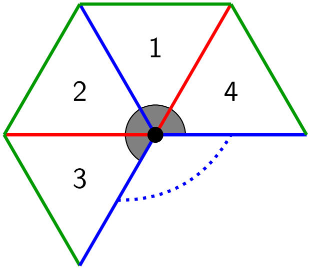
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



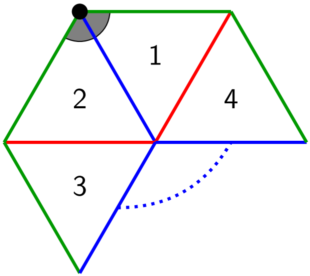
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



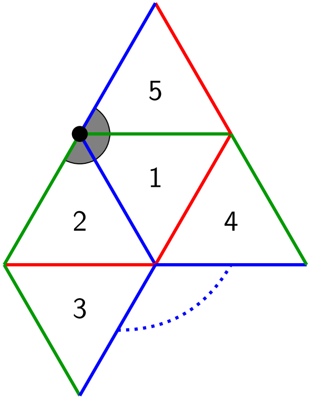
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



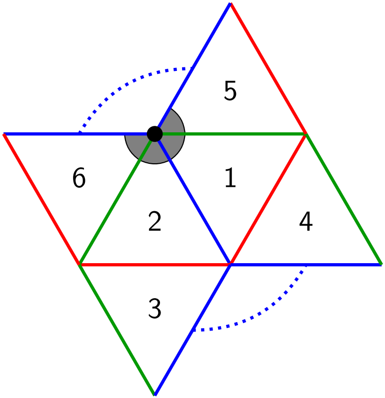
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



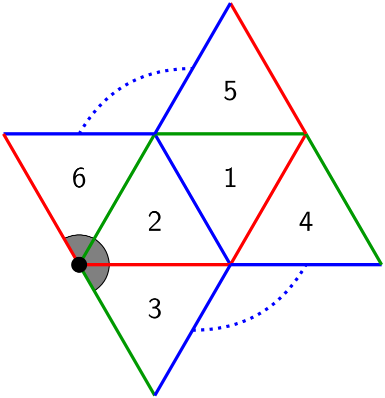
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



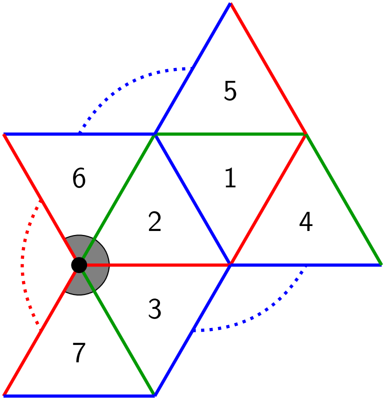
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



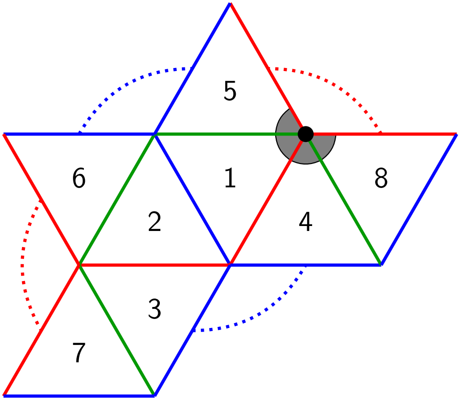
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



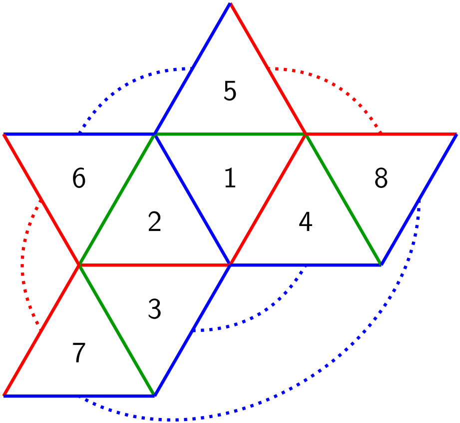
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



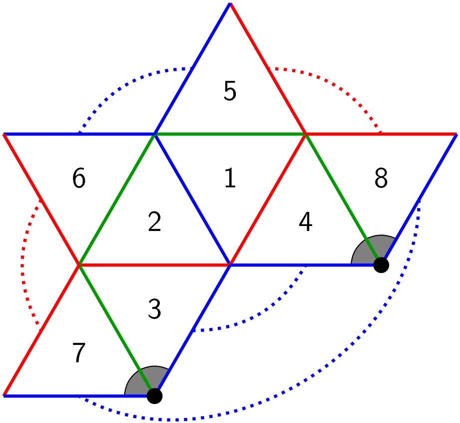
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



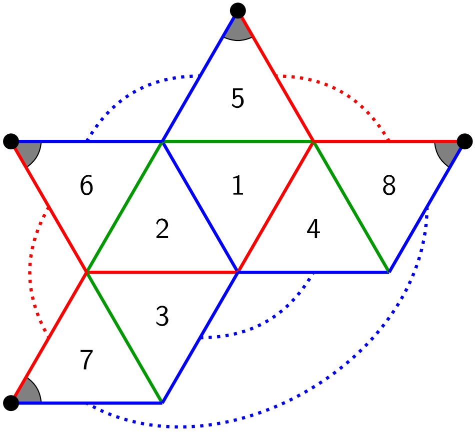
Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



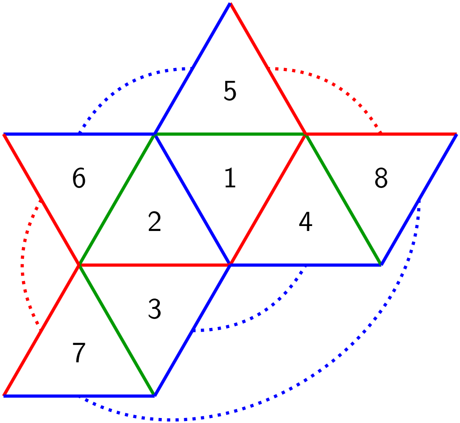
Construction example

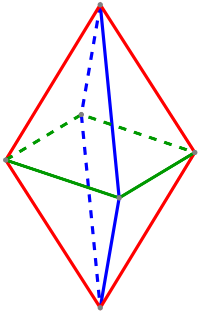
$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$



Construction example

$\sigma_a = (1,2)(3,4)(5,6)(7,8)$
$\sigma_b = (1,4)(2,3)(5,8)(6,7)$
$\sigma_c = (1,5)(2,6)(3,7)(4,8)$





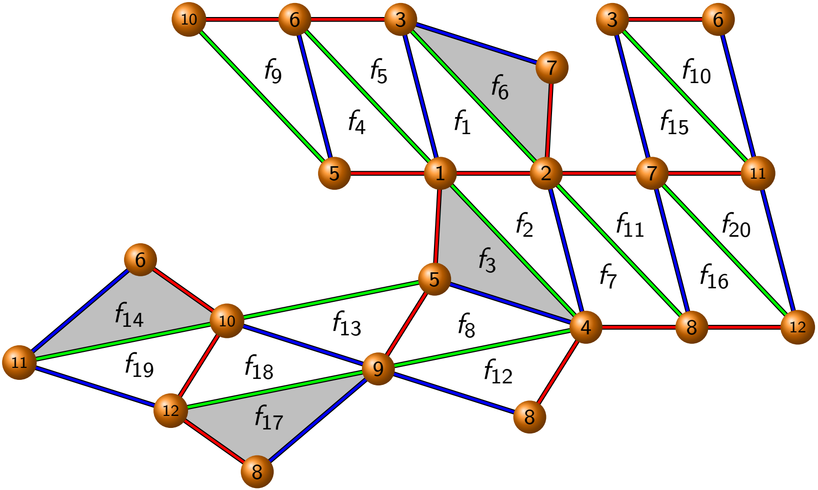
Net of an icosahedron

Net of an icosahedron

```
gap> DrawSurfaceToTikZ(iko,"NetIko.tex");
```

Net of an icosahedron

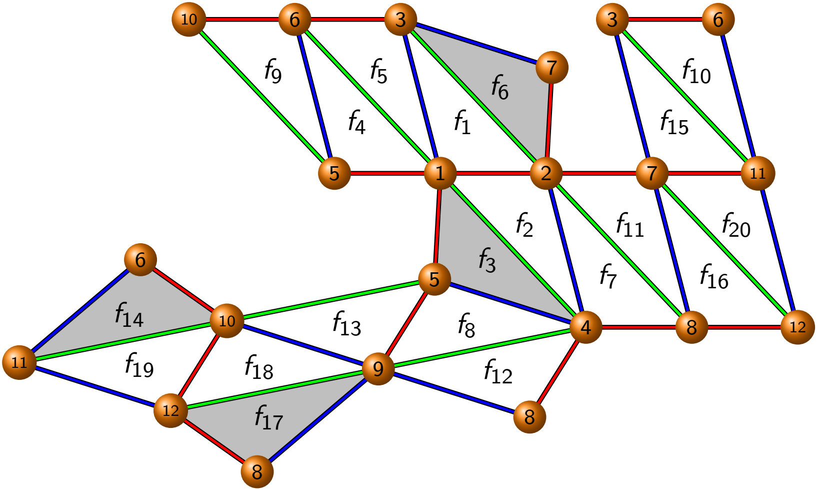
```
gap> DrawSurfaceToTikZ(iko,"NetIko.tex");
```



Net of an icosahedron

```
gap> DrawSurfaceToTikZ(iko,"NetIko.tex");
```

- Has to be manually untangled



Progress report of edge colouring

Progress report of edge colouring

Implemented:

Progress report of edge colouring

Implemented:

- computing all colourings of a given simplicial surface

Progress report of edge colouring

Implemented:

- computing all colourings of a given simplicial surface
- constructing all surfaces with given involutions

Progress report of edge colouring

Implemented:

- computing all colourings of a given simplicial surface
- constructing all surfaces with given involutions
  1. up to (coloured) isomorphism

Progress report of edge colouring

Implemented:

- computing all colourings of a given simplicial surface
- constructing all surfaces with given involutions
  1. up to (coloured) isomorphism
  2. with given mr–assignment

Progress report of edge colouring

Implemented:

- computing all colourings of a given simplicial surface
- constructing all surfaces with given involutions
  1. up to (coloured) isomorphism
  2. with given mr–assignment
- drawing of simplicial surfaces

Progress report of edge colouring

Implemented:

- computing all colourings of a given simplicial surface
- constructing all surfaces with given involutions
  1. up to (coloured) isomorphism
  2. with given mr–assignment
- drawing of simplicial surfaces
- constructing various coloured coverings

Progress report of edge colouring

Implemented:

- computing all colourings of a given simplicial surface
- constructing all surfaces with given involutions
  1. up to (coloured) isomorphism
  2. with given mr–assignment
- drawing of simplicial surfaces
- constructing various coloured coverings

Still missing:

Progress report of edge colouring

Implemented:

- computing all colourings of a given simplicial surface
- constructing all surfaces with given involutions
  1. up to (coloured) isomorphism
  2. with given mr–assignment
- drawing of simplicial surfaces
- constructing various coloured coverings

Still missing:

- for which lengths are the surfaces embeddable?

Progress report of edge colouring

Implemented:

- computing all colourings of a given simplicial surface
- constructing all surfaces with given involutions
  1. up to (coloured) isomorphism
  2. with given mr–assignment
- drawing of simplicial surfaces
- constructing various coloured coverings

Still missing:

- for which lengths are the surfaces embeddable?
- can we predict self–intersections?

Progress report of edge colouring

Implemented:

- computing all colourings of a given simplicial surface
- constructing all surfaces with given involutions
  1. up to (coloured) isomorphism
  2. with given mr–assignment
- drawing of simplicial surfaces
- constructing various coloured coverings

Still missing:

- for which lengths are the surfaces embeddable?
- can we predict self–intersections?
- does varying the lengths lead to another embedding?

What kind of folding?

What kind of folding?

- Folding of surface in $\mathbb{R}^3$

What kind of folding?

- Folding of surface in $\mathbb{R}^3$
- Fold only at given edges (no introduction of new folding edges)

What kind of folding?

- Folding of surface in $\mathbb{R}^3$
- Fold only at given edges (no introduction of new folding edges)
- Folding should be rigid (no curvature)

What kind of folding?

- Folding of surface in $\mathbb{R}^3$
- Fold only at given edges (no introduction of new folding edges)
- Folding should be rigid (no curvature)

Goal: Classify possible folding patterns (given a net)

What kind of folding?

- Folding of surface in $\mathbb{R}^3$
- Fold only at given edges (no introduction of new folding edges)
- Folding should be rigid (no curvature)

Goal: Classify possible folding patterns (given a net)

Embeddings are very hard

Embeddings are very hard

- At every point in time the folding process has to be embedded

Embeddings are very hard

- At every point in time the folding process has to be embedded
- We can only show foldability for specific small examples

Embeddings are very hard

- At every point in time the folding process has to be embedded
- We can only show foldability for specific small examples
  - Usually using regularity (like crystallographic symmetry)

Embeddings are very hard

- At every point in time the folding process has to be embedded
- We can only show foldability for specific small examples
    - Usually using regularity (like crystallographic symmetry)
    - No general method

Embeddings are very hard

- At every point in time the folding process has to be embedded
- We can only show foldability for specific small examples
  - Usually using regularity (like crystallographic symmetry)
  - No general method
- It is very hard to define iterated foldings in an embedding

Embeddings are very hard

- At every point in time the folding process has to be embedded
- We can only show foldability for specific small examples
  - Usually using regularity (like crystallographic symmetry)
  - No general method
- It is very hard to define iterated foldings in an embedding

Folding without embedding

Folding without embedding

Central idea:

Folding without embedding

Central idea:

- Don't model the folding process (needs embedding)

Folding without embedding

Central idea:

- Don't model the folding process (needs embedding)
- Describe starting and final folding state

Folding without embedding

Central idea:

- Don't model the folding process (needs embedding)
- Describe starting and final folding state
    - Only consider changes in the topology

Folding without embedding

Central idea:

- Don't model the folding process (needs embedding)
- Describe starting and final folding state
    - Only consider changes in the topology (like identification of faces)

Folding without embedding

Central idea:

- Don't model the folding process (needs embedding)
- Describe starting and final folding state
  - Only consider changes in the topology (like identification of faces)
  - allows abstraction from embedding

Folding without embedding

Central idea:

- Don't model the folding process (needs embedding)
- Describe starting and final folding state
  - Only consider changes in the topology (like identification of faces)
  - allows abstraction from embedding

$\rightsquigarrow$ Incidence geometry (triangular complex/simplicial surface)

Folding without embedding

Central idea:

- Don't model the folding process (needs embedding)
- Describe starting and final folding state
  - Only consider changes in the topology (like identification of faces)
  - allows abstraction from embedding

$\rightsquigarrow$ Incidence geometry (triangular complex/simplicial surface)

- Captures some folding restrictions (rigidity of tetrahedron)

Folding without embedding

Central idea:

- Don't model the folding process (needs embedding)
- Describe starting and final folding state
  - Only consider changes in the topology (like identification of faces)
  - allows abstraction from embedding

$\rightsquigarrow$ Incidence geometry (triangular complex/simplicial surface)

- Captures some folding restrictions (rigidity of tetrahedron)
- Has to be refined

More than a triangular complex

More than a triangular complex

- Concept should allow reversible folding

More than a triangular complex

- Concept should allow reversible folding
- We need an ordering of the faces:

More than a triangular complex

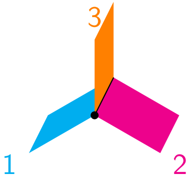
- Concept should allow reversible folding
- We need an ordering of the faces:

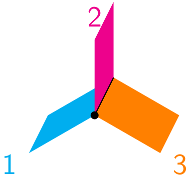
- We also need a cyclic order of the faces around an edge:

More than a triangular complex

- Concept should allow reversible folding
- We need an ordering of the faces:

- We also need a cyclic order of the faces around an edge:

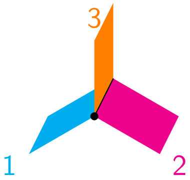


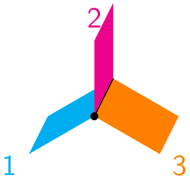


More than a triangular complex

- Concept should allow reversible folding
- We need an ordering of the faces:

- We also need a cyclic order of the faces around an edge:





$\rightsquigarrow$ **folding complex**

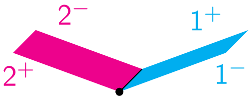
How to describe folding?

How to describe folding?

Needs specification of two face sides:

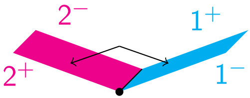
How to describe folding?

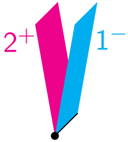
Needs specification of two face sides:



How to describe folding?

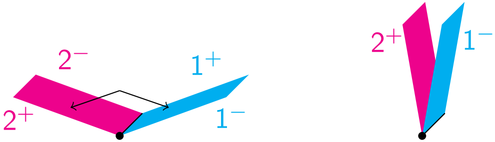
Needs specification of two face sides:





How to describe folding?

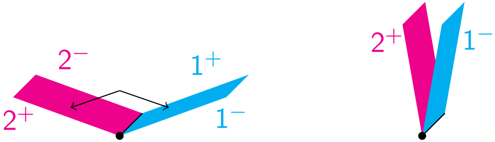
Needs specification of two face sides:



$\Rightarrow$ Characterize folding by specifying the two face sides that touch

How to describe folding?

Needs specification of two face sides:



$\Rightarrow$ Characterize folding by specifying the two face sides that touch
$\rightsquigarrow$ **folding plan**

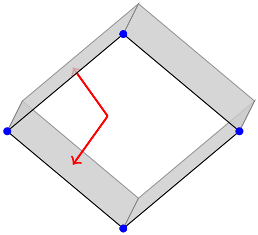
How does folding plan work?

How does folding plan work?

Folding of two faces can induce folding of other faces:

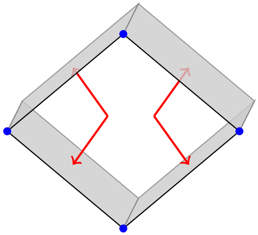
How does folding plan work?

Folding of two faces can induce folding of other faces:



How does folding plan work?

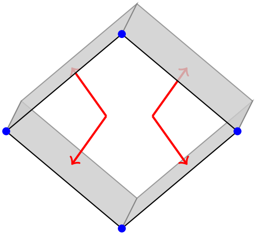
Folding of two faces can induce folding of other faces:



How does folding plan work?

Folding of two faces can induce folding of other faces:

- Can apply to arbitrarily many faces



How does folding plan work?

Folding of two faces can induce folding of other faces:

- Can apply to arbitrarily many faces

How does folding plan work?

Folding of two faces can induce folding of other faces:

- Can apply to arbitrarily many faces

How does folding plan work?

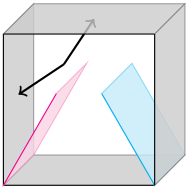
Folding of two faces can induce folding of other faces:

- Can apply to arbitrarily many faces
- The induced folding is not unique

How does folding plan work?

Folding of two faces can induce folding of other faces:

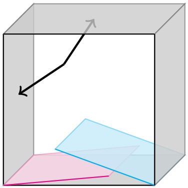
- Can apply to arbitrarily many faces
- The induced folding is not unique



How does folding plan work?

Folding of two faces can induce folding of other faces:

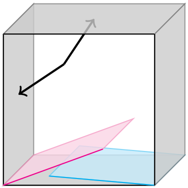
- Can apply to arbitrarily many faces
- The induced folding is not unique



How does folding plan work?

Folding of two faces can induce folding of other faces:

- Can apply to arbitrarily many faces
- The induced folding is not unique

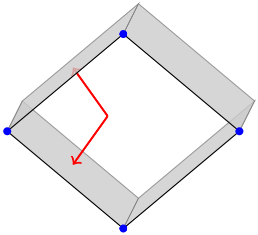


How does folding plan work?

Folding of two faces can induce folding of other faces:

- Can apply to arbitrarily many faces
- The induced folding is not unique

$\Rightarrow$ Identify only two faces at a time

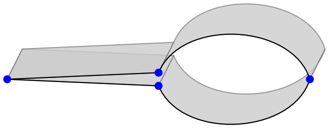


How does folding plan work?

Folding of two faces can induce folding of other faces:

- Can apply to arbitrarily many faces
- The induced folding is not unique

$\Rightarrow$ Identify only two faces at a time



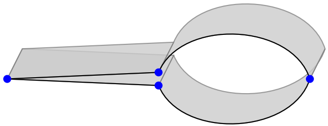
How does folding plan work?

Folding of two faces can induce folding of other faces:

- Can apply to arbitrarily many faces
- The induced folding is not unique

$\Rightarrow$ Identify only two faces at a time

$\rightsquigarrow$ Relax the rigidity–constraint:



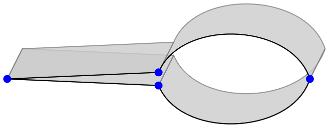
How does folding plan work?

Folding of two faces can induce folding of other faces:

- Can apply to arbitrarily many faces
- The induced folding is not unique

$\Rightarrow$ Identify only two faces at a time

- $\rightsquigarrow$ Relax the rigidity–constraint:
  - Allow non–rigid configurations as transitional states



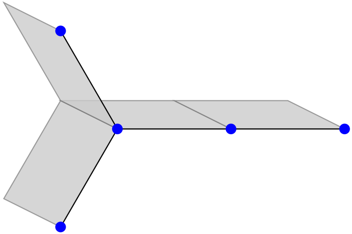
Structure of multiple foldings

Structure of multiple foldings

With folding plans we can perform the same folding in different folding complexes

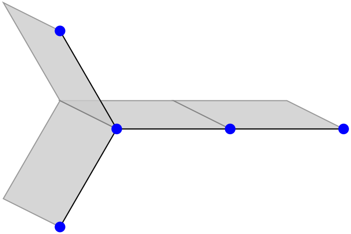
Structure of multiple foldings

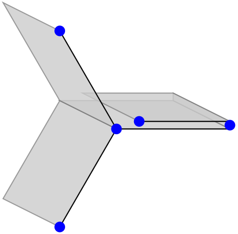
With folding plans we can perform the same folding in different folding complexes



Structure of multiple foldings

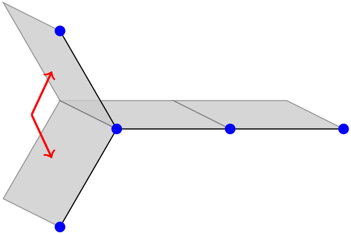
With folding plans we can perform the same folding in different folding complexes

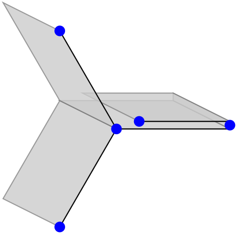




Structure of multiple foldings

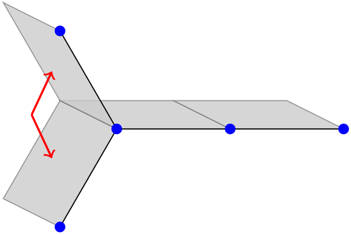
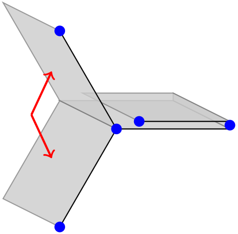
With folding plans we can perform the same folding in different folding complexes





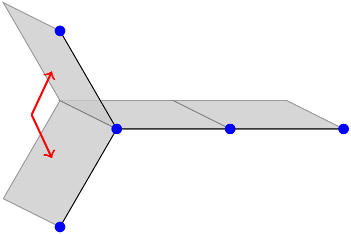
Structure of multiple foldings

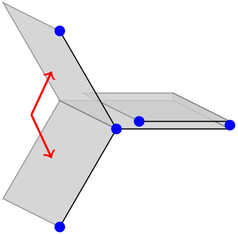
With folding plans we can perform the same folding in different folding complexes

Structure of multiple foldings

With folding plans we can perform the same folding in different folding complexes





$\rightsquigarrow$ more structure on the set of possible foldings

Folding graph

Folding graph

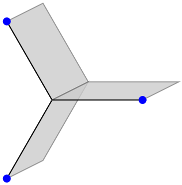
- Vertices are folding complexes (modelling folding states)

Folding graph

- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes

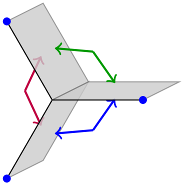
Folding graph

- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes



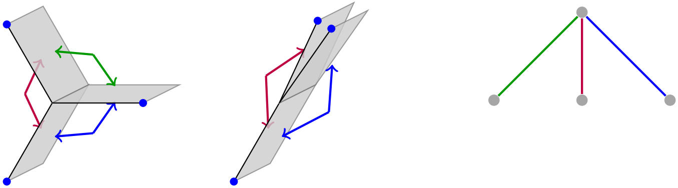
Folding graph

- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes



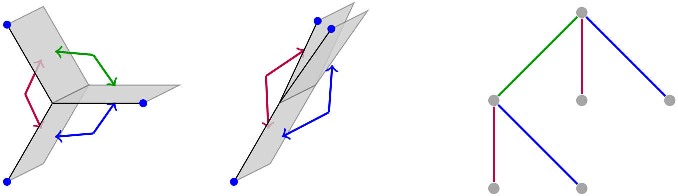
Folding graph

- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes



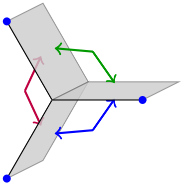
Folding graph

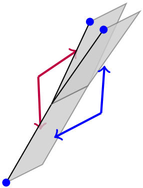
- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes

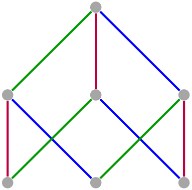


Folding graph

- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes

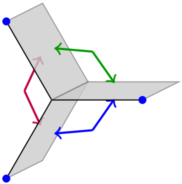


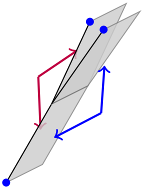


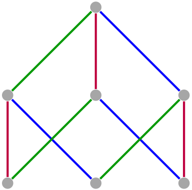


Folding graph

- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes

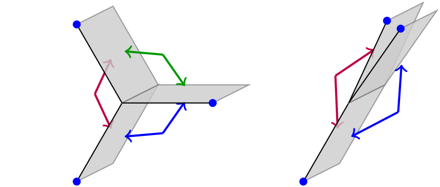


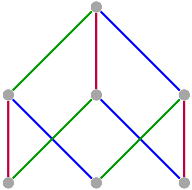


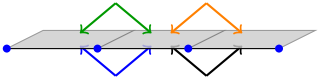


Folding graph

- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes

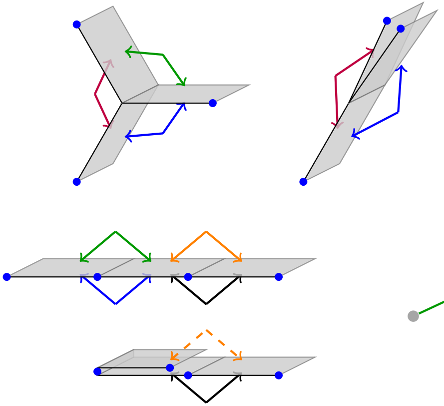


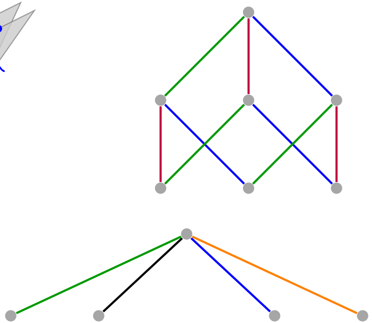




Folding graph

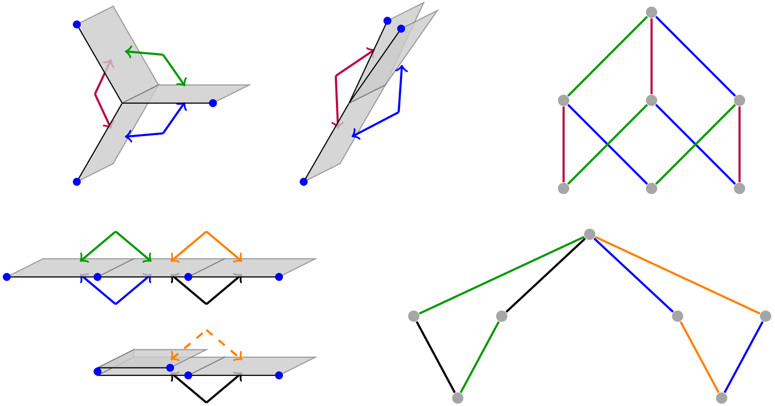
- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes





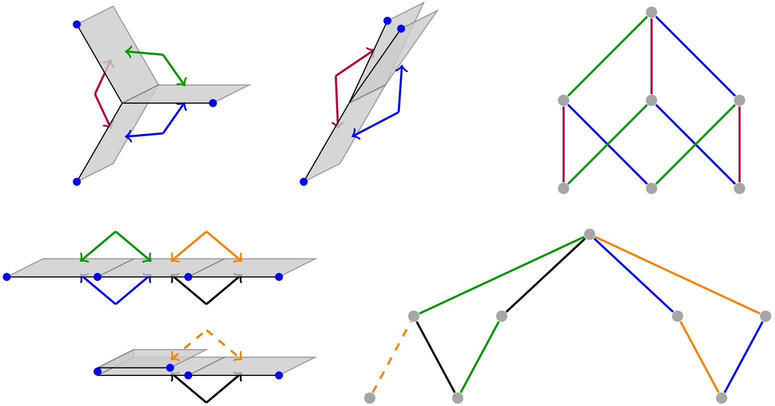
Folding graph

- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes



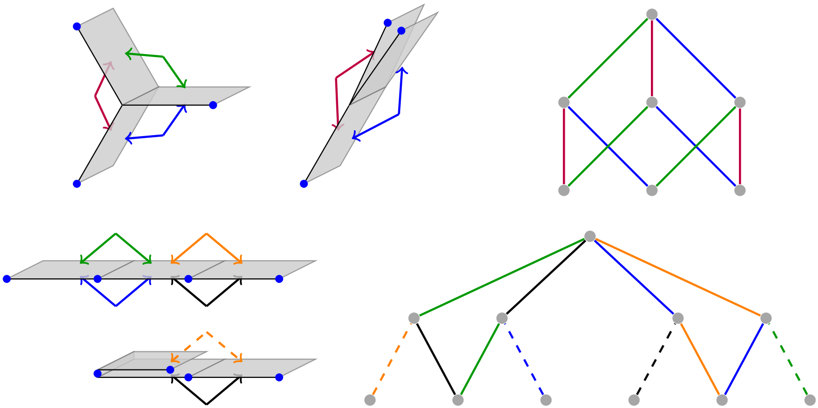
Folding graph

- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes



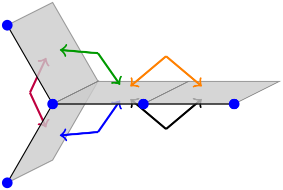
Folding graph

- Vertices are folding complexes (modelling folding states)
- Edges are folding plans connecting two folding complexes

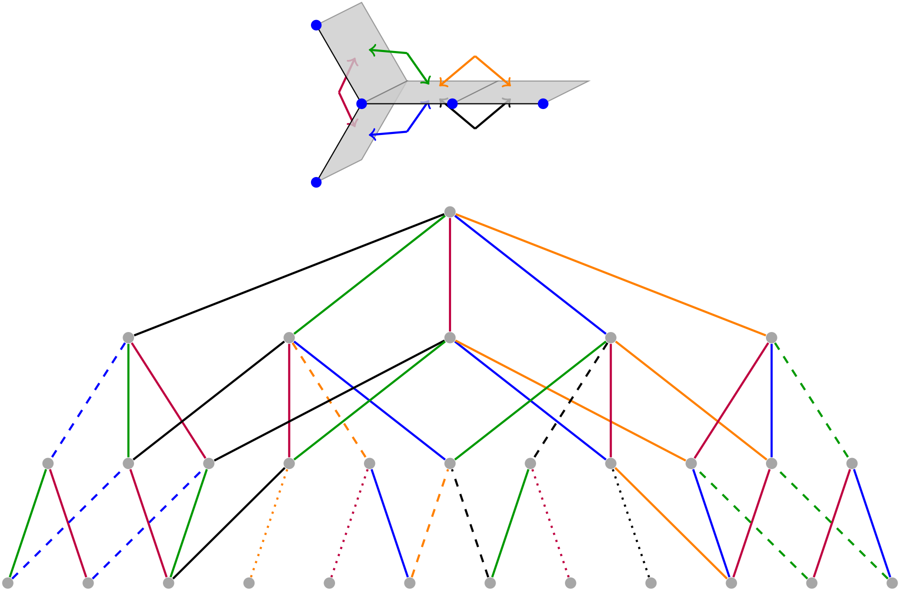


Larger graph

Larger graph



Larger graph



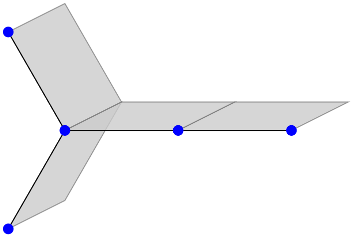
Drawback of folding plans

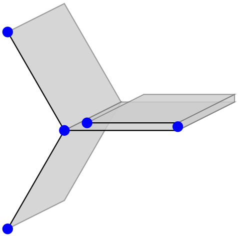
Drawback of folding plans

Some foldings that “should” be the same, aren’t:

Drawback of folding plans

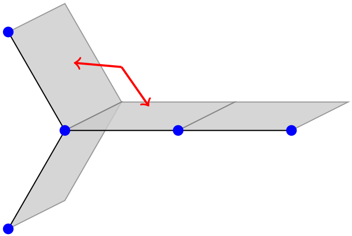
Some foldings that “should” be the same, aren’t:

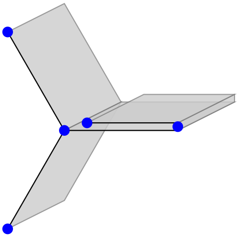




Drawback of folding plans

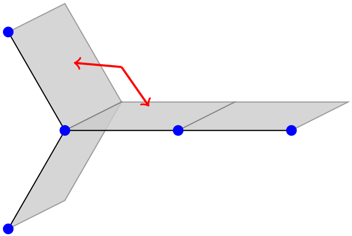
Some foldings that “should” be the same, aren’t:

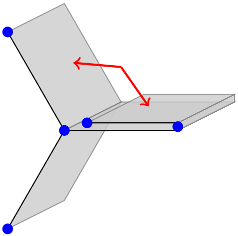




Drawback of folding plans

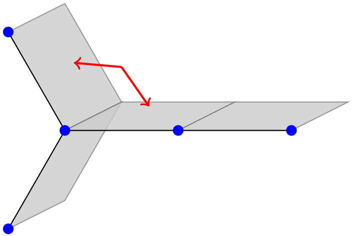
Some foldings that “should” be the same, aren’t:

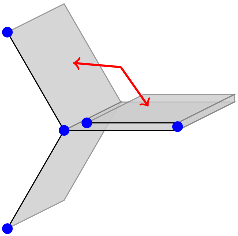




Drawback of folding plans

Some foldings that “should” be the same, aren’t:

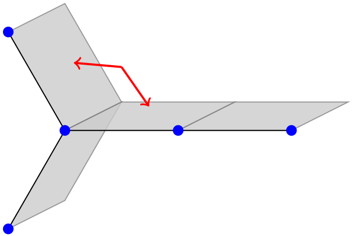


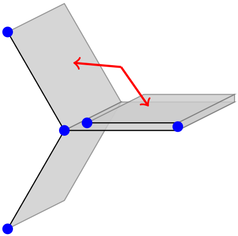


$\Rightarrow$ If you know the folding structure of a small complex,

Drawback of folding plans

Some foldings that "should" be the same, aren't:

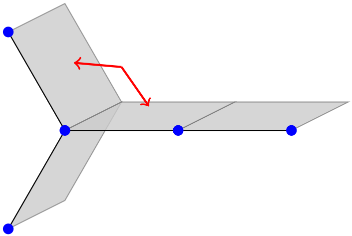


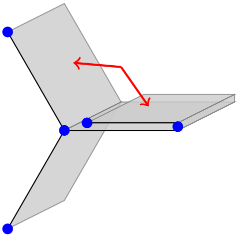


$\Rightarrow$ If you know the folding structure of a small complex, you can't easily find the folding structure of an extended complex

Drawback of folding plans

Some foldings that “should” be the same, aren’t:





$\Rightarrow$ If you know the folding structure of a small complex, you can’t easily find the folding structure of an extended complex

$\rightsquigarrow$ Folding plans are not optimal to model folding

Progress report of abstract folding

Progress report of abstract folding

In development:

Progress report of abstract folding

In development:

- folding complex

Progress report of abstract folding

In development:

- folding complex
- folding plans

Progress report of abstract folding

In development:

- folding complex
- folding plans
- folding graph

Progress report of abstract folding

In development:

- folding complex
- folding plans
- folding graph

Missing:

Progress report of abstract folding

In development:

- folding complex
- folding plans
- folding graph

Missing:

- better folding description

Progress report of abstract folding

In development:

- folding complex
- folding plans
- folding graph

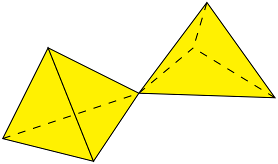
Missing:

- better folding description
- properties of folding graphs

Summary `SimplicialSurfaces`

Summary SimplicialSurfaces

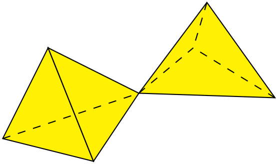
Triangulated complexes



Summary `SimplicialSurfaces`

Triangulated complexes

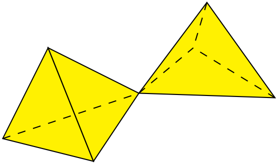
- mostly complete



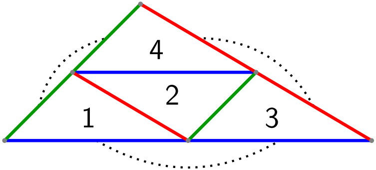
Summary SimplicialSurfaces

Triangulated complexes

- mostly complete



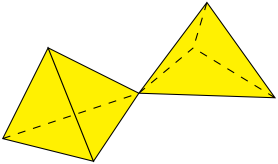
Edge colouring



Summary `SimplicialSurfaces`

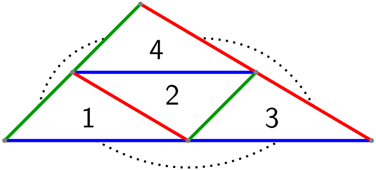
Triangulated complexes

- mostly complete



Edge colouring

- current theory implemented



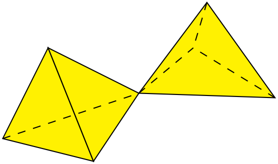
Summary SimplicialSurfaces

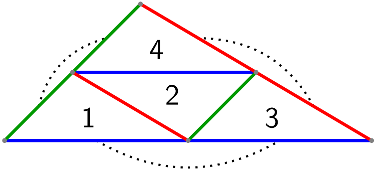
Triangulated complexes

- mostly complete

Edge colouring

- current theory implemented
- a lot of theory missing

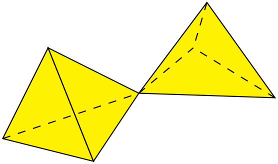




Summary SimplicialSurfaces

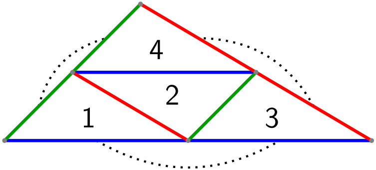
Triangulated complexes

- mostly complete

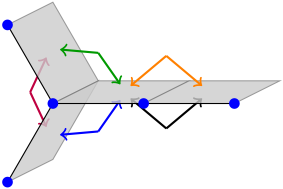


Edge colouring

- current theory implemented
- a lot of theory missing



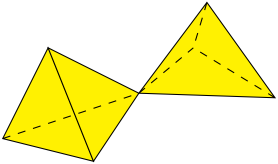
Abstract folding



Summary `SimplicialSurfaces`

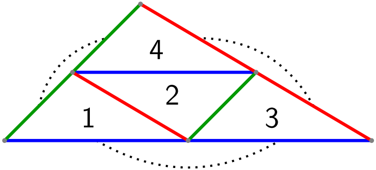
Triangulated complexes

- mostly complete



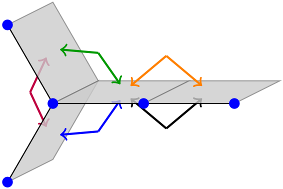
Edge colouring

- current theory implemented
- a lot of theory missing



Abstract folding

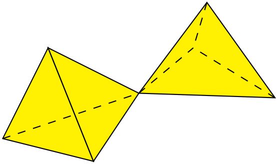
- framework exists



Summary SimplicialSurfaces

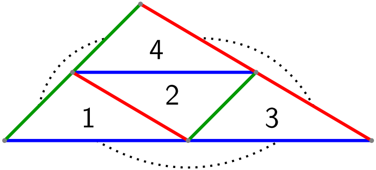
Triangulated complexes

- mostly complete



Edge colouring

- current theory implemented
- a lot of theory missing



Abstract folding

- framework exists
- needs proper implementation